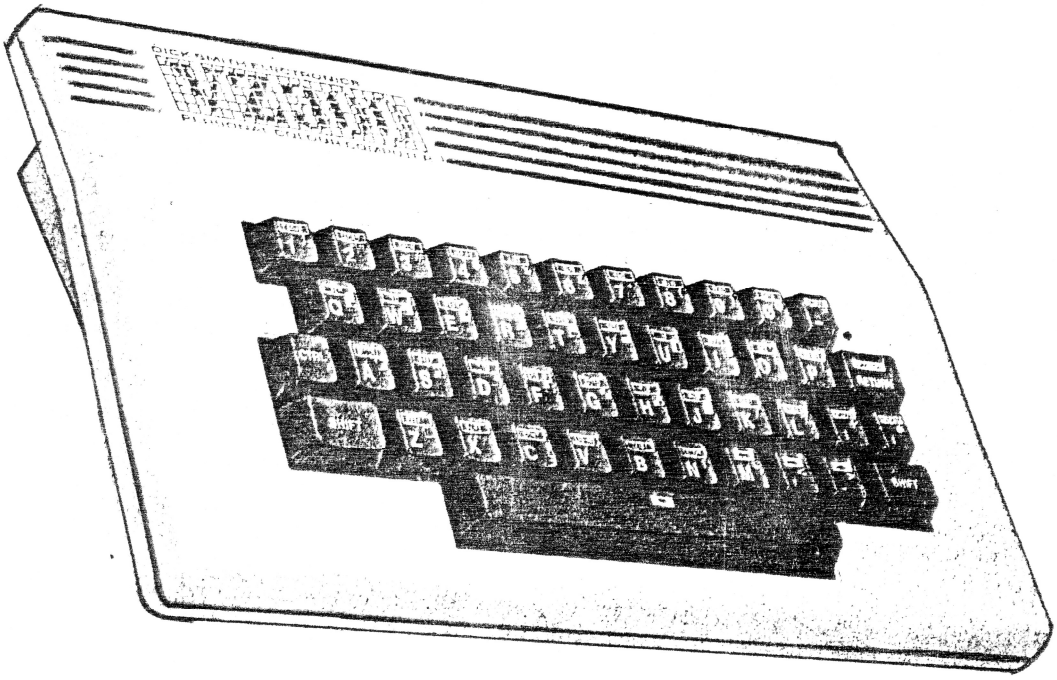
VZ_LINK

NEWSLETTER



AUCKLAND VZ 300 / 200
COMPUTER USERS CLUB
P.O BOX 1972 c.p.o

ISSUE No: 7 APRIL 87

(C) Auckland VZ 300/ 200 computer user's club 1986.

All material in this newsletter is subject to copyright. Contributed program's & articles can be reproduced on the under standing that they are for the private use of subscriber's only. Copyright is retained by the author.

CONTRIBUTIONS

The following points will help those thinking of sending program's for publication in this newsletter.

If possible send a listing and a tape of your program. This helps me prepare the newsletter and the tape will be returned when I have finished.

Include some notes on what the program does and how to use it.
What is painfully obvious to the author is often obscure to we users.

For long or complex program's let me know if I can supply readers with a copy of it on tape or if you are prepared to do this yourself.
I urgently require program's and items for this newsletter.

Membership $10-00 per year for 12 issues

POSTAL ADDRESS

P.O Box 1972 C.P.O AUCKLAND...

NEWS & VIEWS APRIL 87.

Well it's less than a fortnight before I head of to Christchurch. I am really looking forward to visiting with the Christchurch users.

I am also looking forward to seeing what the D.S.E store have to offer.

In the March newsletter I made a mistake when announceing the six new VZ programs and then went on to list only five. The sixth program is of course " Space Ram ", A review of "Space Ram" appeared in the March newsletter.

I have been giving a lot of thought to forming a software and program library for the VZ. I have a lot of VZ software and I am sure other users do to.

It would be great to have a central software bank that users can use.

This would enable users to view a program before purchase or just save money.

You would be able to use the program or play the game then return it. The idea is sound, maybe some of you have other idea's?. Let's hear them, put pen to paper and send them to me. It's time we got organised before all these programs are lost for ever.

I would also like to get a top game score list going but I, am not sure how this would work, again any idea's on this, Let me know. I am still begging for programs hints or any item's for the newsletter. Please, Please help.

In the May newsletter I will again, be doing user survey just to make sure all users are satisfied with the newsletter content. You can help by filling the survey in and returning same to me.

In this issue, I have included another technical bulletin. I feel we must reprint.

As far as I know these bulletins have never been released in N.Z. Also in this issue is my review of the new Luner Lander program. I write these reviews so that users know what a program is all about and don't buy totally useless programs. If after reading the reviews you are still not sure, contact me and we can arrange for you to see the program in action.

Attached to the rear of this newsletter is a page from Bob Kitchs item. Which I have been running in the last few news letter. Some how this page was missed and should be in issue 6 btwn page 4 &5. I am sorry this happened, and I hope it didn't inconvenience anyone.

This page has been written using the tp-40 printer plotter and wordprocessor

The following hints have come to light:-

There are two versions of the disc controller that i know of to date.
v1,1 and v1,2 even at this early stage the differences are showing up.
The "DIR" does not seem to work on the v1.1.
please let me know of any quirks you are having.

When <csave>ing programmes.enter the following in the immediate mode ie without a line number,then<return>.
csave"your prog name ":for x=1to20:sound30,1:next:verify
This <csave>.s your programme,gives 20 beeps when saved,and put's the computer in the <verify>mode.
Just rewind the tape and play.

When using a disc drive unit,ALWAYS TURN THE DISC DOOR KNOB TO OPEN BEFORE SWITCHING THE COMPUTER AND/OR DISC DRIVE ON/OFF .
There are times when all your disc data,or access to it is lost if you do not do this.
You can even lose all the programme in ram.

This short m/l programme LPRINTS the disc drive directory.

```
001 ; directory printout
002 ; load printer device code
003        ld          A,1
004        ld          (789Ch),A
005 ; call dir routine
006        call 4906h
007        ret
```

In one of the technical bulletins no 93. It was stated that to get the line-up code to work on the tp-40 you use OUT 1,11
This does not work but if you use OUT 12,11:OUT 13,0
Be carefull how you use this as the buffer is not checked before hand

MOST DICK SMITH STORES IN AUCKLAND NOW HAVE MOST OF THE VZ SOFTWARE AVAILABLE INCLUDING ED/ASS. AND SPRITE PACKAGE

Tape Loading Format

	T: Text File	B: Binary File	D: Data File
SYNC. Bytes	255 Bytes of 80H	255 Bytes of 80H	255 Bytes of 80H
HEADER	5 Bytes of FEH	5 Bytes of FEH	5 Bytes of FEH
EXTENSION	1 Byte of F0H	1 Byte of F1H	1 Byte of F2H
FILENAME	16 Bytes (max.) of ASCII	16 Bytes (max.) of ASCII	16 Bytes (max.) of ASCII
GAP	3 ms Blank	3 ms Blank	3 ms Blank
START ADDRESS	2 Bytes of binary	2 Bytes of binary	----
END ADDRESS	2 Bytes of binary	2 Bytes of binary	----
Program Content	xx Bytes	xx Bytes	----
Data Content	----	----	xx Bytes
Checksum	2 Bytes	2 Bytes	2 Bytes
End of File Marker (EOF)	20 Bytes of Zeroes (00H)	20 Bytes of Zeroes	----
Terminator	----	----	1 Byte of 00H

```
5 REM THIS WAS SENT IN BY ANDREW WILLOWS
6 REM HOLD DOWN DIFFERENT KEYS
10 MODE(1)
20 POKE28670,12:POKE28670,16
30 GOTO20
```

Dick Smith Electronics

Technical Bulletin

FINDING WHERE VARIABLES ARE STORED BY THE VZ-200'S BASIC

When programming the VZ-200 computer in BASIC, there are times when you need to know where the interpreter has stored your program's variables. Superficially this is not easy, as the VZ-200's BASIC seems to have no VARPTR function. However as it happens the VARPTR routine is actually present inside the VZ-200's ROMs, even though the input/tokenising section of the interpreter cannot recognise the keyword 'VARPTR' and turn it into the appropriate token.

To use the routine simply, it is possible to 'trick' the interpreter by POKEing the appropriate token (C0 hex or 192 decimal) into a program line, in place of a similar token -- say that for USR (C1 hex or 193 decimal). As the execute section of the interpreter can treat the VARPTR token normally and call the appropriate routines, this gives the desired result when the program is RUN. However because the LIST and LLIST routines cannot recognise the VARPTR token, the line with this token in cannot be listed properly.

Here is a small sample program which should illustrate how the above 'trickery' can be performed from within your BASIC program itself -- in this case for a string variable:

```
 10 GOTO30
 20 X=USR(A$):RETURN
 30 D=PEEK(30864)+256*PEEK(30885):REM FINDS START OF PROG
 40 B=PEEK(D):IFB<>193THEND=D+1:GOTO40:REM FIND 'USR' TOKEN
 50 IFB=193THENB=192:POKED,B:REM & REPLACE WITH 'VARPTR' TKN
 60 REM NOW TRY IT OUT
 70 A$="WHATEVER":GOSUB20:REM GO FIND PTR FOR A$
 80 FOR I=XTOX+2:LPRINTI,PEEK(I):NEXTI:REM PRINT PTRS OUT
 90 PT=PEEK(X+1)+256*PEEK(X+2):REM NOW SET PT FOR A$ START
100 FOR I=PT TO PT+PEEK(X)-1
110 LPRINTCHR$(PEEK(I));
120 NEXTI:LPRINT
```

The actual line which eventually calls the VARPTR routine is line 20, which is placed as near as possible to the start of the program so that it can be located easily to swap tokens.

.../2

FINDING WHERE VARIABLES ARE STORED BY THE VZ-200'S BASIC -- 2

As you can see it is made a subroutine, so that the VARPTR function can effectively be called from anywhere in the program using GOSUB20. Line 10 is simply arranged to skip over the subroutine to line 30, the 'real' start of the program proper.

Lines 30 to 50 perform the actual token swapping in line 20. First line 30 finds the pointer to the start of BASIC's program storage area. Then line 40 examines each byte in memory, starting at the beginning of the program, until it finds the USR token (decimal 193). Then line 50 pokes the VARPTR token code back into the same address. So after these lines are RUN, line 20 will behave as if it were written:

```
20 X=VARPTR(A$):RETURN
```

Lines 70 to 120 are to demonstrate how the subroutine works. First, line 70 gives string variable A$ a value, then calls the subroutine so that X will be given the pointer value for it. Then line 80 prints out the string's length and storage address bytes. Finally line 90 sets PT to point to the actual string storage address, and lines 100-120 read it and print it out.

If you RUN this program, this is what you get:

```
 31701          8
 31702          86
 31703          123
WHATEVER
```

As you can see, the first byte stored at the pointer address for a string variable is the length of the string. The next two bytes form the pointer to where the string is actually stored.

Note that if you try to LIST or LLIST the above program <u>after</u> it has been run, line 20 will look like this:

```
20 X=_
```

As noted earlier, this is simply because the listing routines cannot identify the VARPTR token.

.../3

FINDING WHERE VZ-200'S BASIC STORES ITS VARIABLES -- 3

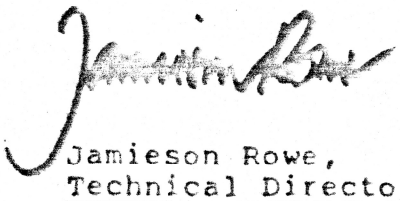
Needless to say, this approach isn't confined to finding string variables. It can be used for each type of variable, although line 20 will obviously need to be changed to suit the type of variable involved. For example if you use X=USR(A) it would be suitable for either integer or single-precision numeric variables.

Here is the format used for the pointers retrieved for the various types of variable (X = first pointer location):

INTEGER VARIABLE:	X	contains LSB of variable itself
	X+1	contains MSB
SINGLE PREC. VBL:	X	contains LSB of variable itself
	X+1	contains next most sign. byte
	X+2	contains MSB
	X+3	contains exponent of value
DOUBLE PREC. VBL:	X	contains LSB of variable itself
	X+1	contains next most sign. byte
	X+2	contains next most sign. byte
	X+3	contains MSB
	X+4	contains exponent of value
STRING VARIABLE:	X	contains length of string
	X+1	contains LSB of string's start
	X+2	contains MSB of string's start

By modifying the above technique slightly, you could have two or more 'VARPTR' routines, one to suit each type of variable your program needs to find.

Regards,



Jamieson Rowe,
Technical Director

REVIEW OF D.S.E Program LUNAR LANDER

At first i was not going to purchase this program .
I already had a program called Lunar Lander which was put out by Dick smiths sometime back for the VZ.

As often happens i saw the new lunar lander program running in one of the stores.

I realized that although the name and idea was the same,The program was all new!!.

Boy, was i disappointed .Not only was this game hard to play but i had no idea what was meant to happen.
Unfortunately the instructions seemed to be less than clear.

You have the control of the lander and must land it safely, Nothing hard in that you may say,but you just try it and see.

you control the main and left/right thrusters as the lander comes down.

I found the keyboard or joystick control's very unresponsive. Maybe it was just me,but the side thrusters didn't seem to have much effect on movement when you needed it.

THe other thing that i found missing was sound.There is no sound what so ever,not even a beep.

The VZ graphics were used very well and it sure seems a shame not to add some sound.

On looking through a micro 80 mag,i found a adv. for a Lunar lander game which was for a Trs 80 and it even had sound.

This was back in 1982 ,so i wonder what happened in the VZ version.

Sadly LUNAR LANDER is just one more in a seemingly inexhaustable line of many similar games.

I would like very much to see more programs without the space scenario as their background .

D.S.E would be better of reviving the old lunar lander program and updating it for resale.

After the first few attempts to play this game ,i gave up and put the program away.

By PJ HILL
(AUCKLAND NZ)

AUCKLAND VZ300/200 COMPUTER USERS CLUB

P.O BOX 1972 C.P.O AUCKLAND

MEMBERSHIP REQUEST

NAME

PHONE

ADDRESS

MODEL vz300 or 200

AGE GROUP under20 20%40 over40

EXTRAS(ram?/disk drive etc)

WHICH OF THE FOLLOWING BEST DESCRIBES YOU (tick)

-computer beginner
-computer hobbiest
-electronics hobbiest
-computer technician
-other _ -

DO YOU HAVE ANY PROGRAM'S OR OTHER ITEM'S THAT YOU WOULD BE WILLING TO SHARE WITH OTHER USERS ? (y/n)

COMMENTS

(if you require n/letter back issues)
indicate numbers here ----------

MEMBERSHIP FEE

$10-00 per year (12 issues of newsletter)

BACK ISSUES AVAILABLE AT $1.00 per issue to NON* MEMBERS

NEW MEMBERS WILL RECEIVE BACK ISSUES(IF AVAILABLE)FREE ON REQUEST

PLEASE MAKE CHEQUES PAYABLE TO:

VZ 300/200 COMPUTER CLUB

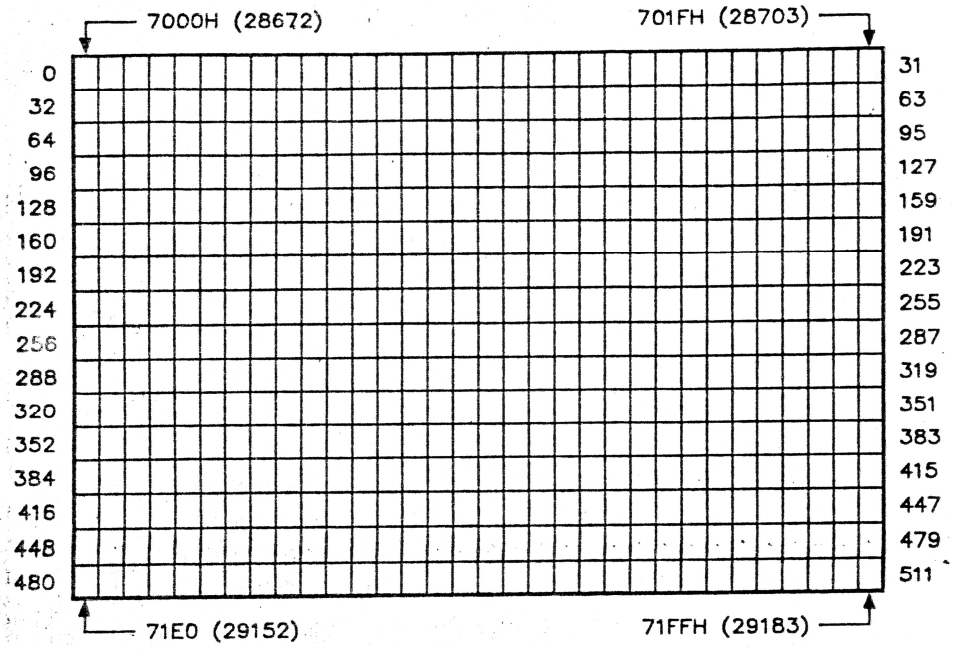


Figure 5. Screen addressing for MODE (0) or lo-res displays on VZ computers. This mode corresponds to Alphanumeric and Semigraphic 4 on VDG and is 32 by 16 characters in size. Each character is byte-mapped as indicated.

being selectable from green or orange. Remember that the Inverse mode of the MC6847 is set by bit 6 of the data value contained in video RAM. (see also Figures 1 and 6).

An understanding of this involves looking at individual bits within the bytes and also looking at how these bits can control and reset certain control lines on the VDG (as outlined in Part 1).

In text mode there are 64 characters in each of the Normal (0-63) and Inverse (64-127) sets. This implies that a 6-bit code is used to encode the character shape and that bit 6 determines whether Normal or Inverse.

For example:—

b7	b6	b5	b4	b3	b2	b1	b0	
0	0	1	0	0	1	0	1	Binary = 37D or '%' normal.
0	1	1	0	0	1	0	1	Binary = 101D or '%' inverse.

Note the way that bit 6 determines normal/inverse. Also note that bit 7 does not change. The most significant bit (MSB) is used to indicate text character to the on-chip ROM.

In summary, for the character source, a 6-bit ASCII code is used to call the elemnent from the on-chip ROM, the seventh bit indicates normal or inverse illumination, and the eighth bit is held low to indicate Alphanumeric mode.

b7	b6	b5	b4	b3	b2	b1	b0
alpa *A/S	inv INV	6-bit			ASCII		

In graphics mode the Semigraphics 4 mode of the VDG is used. The 8 by 12 pixel character is divided into four 'rectangular' quadrants of size 4 by 6. The quadrants are 'psuedo-addressable' by selecting the correct area as shown on Figure 7.

In Semigraphics mode, a more comprehensive form of encoding is used. The character codes extend from 128 to 255, implying that the MSB (or bit 7) is set to 1 (or high) to indicate that a graphics character is encoded in the byte. The graphic block character contains 16 discrete patterns involving 'switching' on or off the four quadrants. The four low-order bits handle a quadrant a piece (refer Figure 7). Additionally one-of-eight illumination colours is encoded in the next three bits (bits 6 to 4).

For example:—

b7	b6	b5	b4	b3	b2	b1	b0	
0	0	1	0	0	1	0	1	Binary = 217D or cyan
0	1	1	0	0	1	0	1	Binary = 145D or yellow

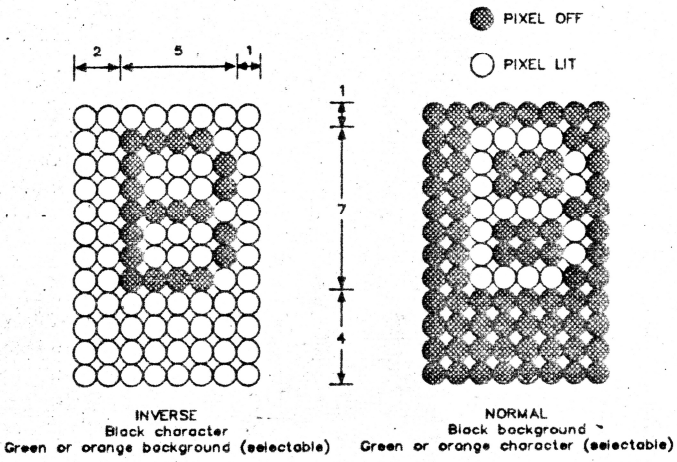
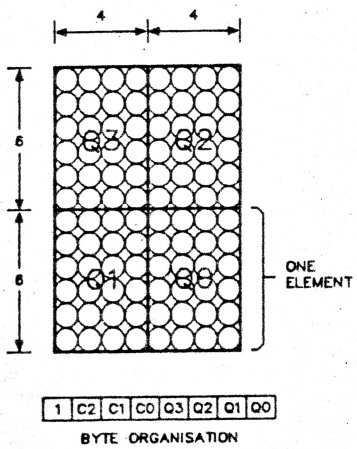


Figure 6. Format of Alphanumeric Mode — Internal on MC6847. Each character is 12 by 8 pixels and each screen is 32 by 16 characters. A 6-bit ASCII code specifies the character from an on-chip ROM.



Qx	C2	C1	C0	COLOUR
0	X	X	X	BLACK
1	0	0	0	GREEN
1	0	0	1	YELLOW
1	0	1	0	BLUE
1	0	1	1	RED
1	1	0	0	BUFF
1	1	0	1	CYAN
1	1	1	0	MAGENTA
1	1	1	1	ORANGE

Figure 7. Format of Semigraphic 4 Mode on MC6847. Each character is 12 by 8 pixels but elements or quadrants can be individually illuminated giving a screen resolution of 64 by 32 elements in up to eight colours.